

A generalized particle transport model for HIGRAD

Robert Martin-Short^{1,2}, Eunmo Koo², Keira Haskins³, Robert Robey⁴

martinshortr@gmail.com

1. University of California Berkeley
2. Los Alamos National Lab, EES-16
3. University of New Mexico
4. Los Alamos National Lab, XCP-2

LA-UR-17-26931

Introduction

The HIGRAD (High-Gradient Atmospheric Model) computational fluid dynamics (CFD) software has been updated to incorporate a new particle transport routine and improvements for efficiency and stability on parallel computing architectures.

HIGRAD uses the finite volume method to solve the compressible, grid-filtered Navier-Stokes equations on a regular, Cartesian grid. It was developed at Los Alamos National Laboratory (LANL) and has been employed to solve a variety of scientific problems in atmospheric hydrodynamics.

We extend this functionality by adding a lagrangian particle tracking capability. Our new module is generalized for the simulation of particles of various types and properties. Thus, it has a wide range of potential applications, from cloud-physics to spotting ignitions by firebrands in wildfires.

We test our implementation through use of a new HIGRAD test case - a two dimensional (2D) idealized explosion incorporating lagrangian particles - and visualize the results in real time using a newly-developed, lightweight graphics package using OpenGL.

The particle tracking module is designed to run efficiently on massively parallel architectures, and is able to take advantage of rank-parallelism (via MPI) and threading (OpenMP).

Applications of HIGRAD



Figure 1: Wildfire simulation, coupled with FIRETEC.

Software has the capability to incorporate topography and varying vegetation types, derived from analysis of satellite imagery (e.g. Koo et al., 2012)

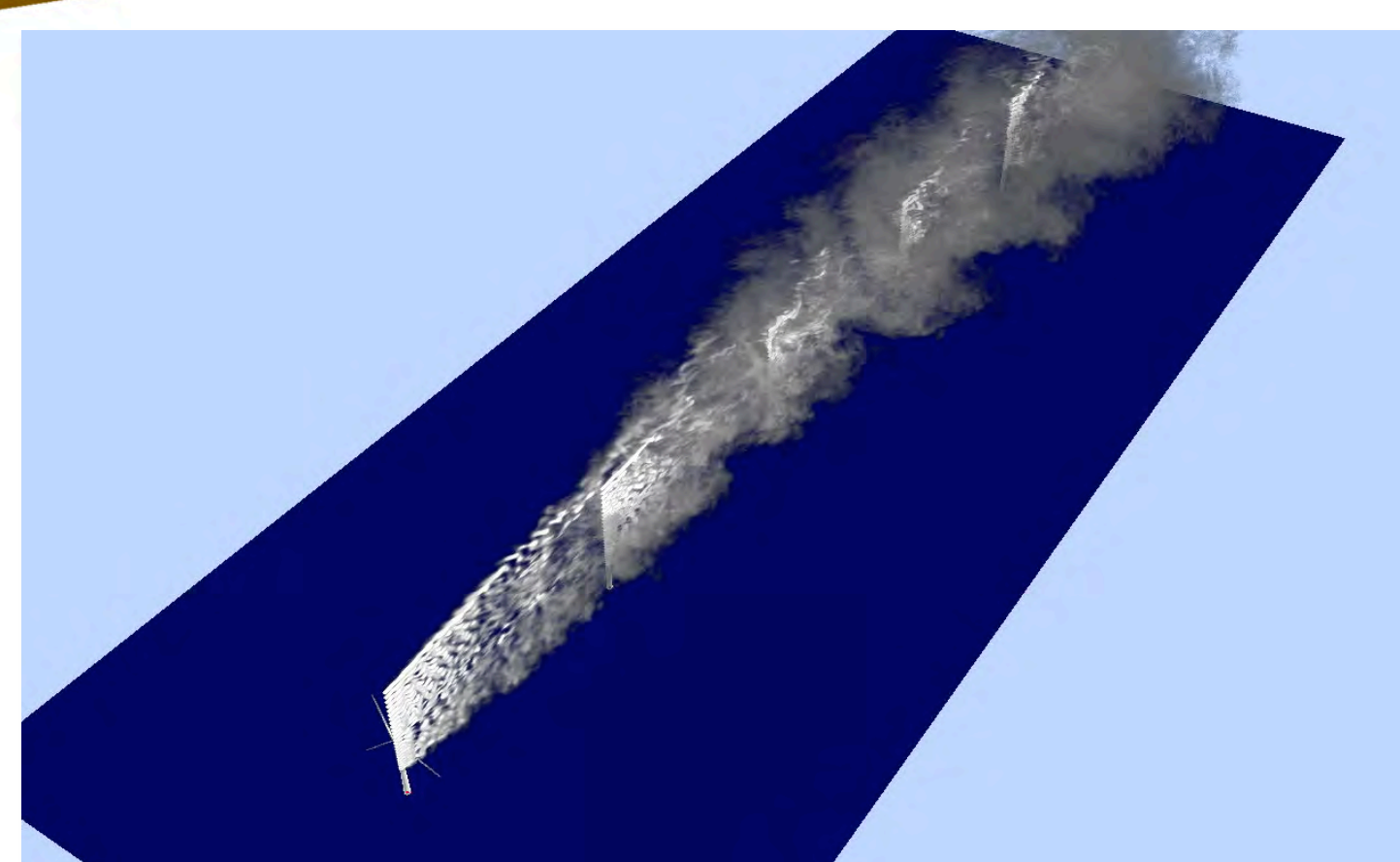


Figure 2: Visualization of the turbulent wake of a wind turbine
A wind turbine and wake model has been developed to model the power output of wind arrays (e.g. Linn et al. 2013)

Background

Lagrangian particle tracking is particularly useful in problems dominated by advection and diffusion effects (Cheng & Plassmann, 2001). Given an Eulerian velocity field, the goal is to trace the time trajectories of tagged fluid particles as the flow evolves (Yeung & Pope, 1988). When designing a parallel algorithm for particle tracking, the following should be considered:

- An appropriate interpolation scheme (typically bilinear)
- A method of communicating particles between processes
- A scheme to update particle positions (typically linear)
- A method of representing and storing the particles in memory
- An efficient algorithm for particle-particle interaction

In the present form, our particles may have varying properties but do not interact. Thus computation of their trajectories is a parallel independent process.

Particle representation

We test two different methods of storing and communicating particle information:

A four-dimensional array structure that stores attributes of the particles present in each grid cell (Particle version 1)

Advantages: A natural way of keeping track of large numbers of particles of different types; can easily handle multiple particles in the same grid cell

Disadvantages: Requires memory allocations/deallocations on transfer of particles between cells; difficult to debug

A one-dimensional array structure containing the particles, attributes of which include the grid cells in which they reside (Particle version 2)

Advantages: Requires fewer memory access operations; conceptually simpler

Disadvantages: May be more difficult/inefficient to implement a particle collision model with this framework

Performance analysis

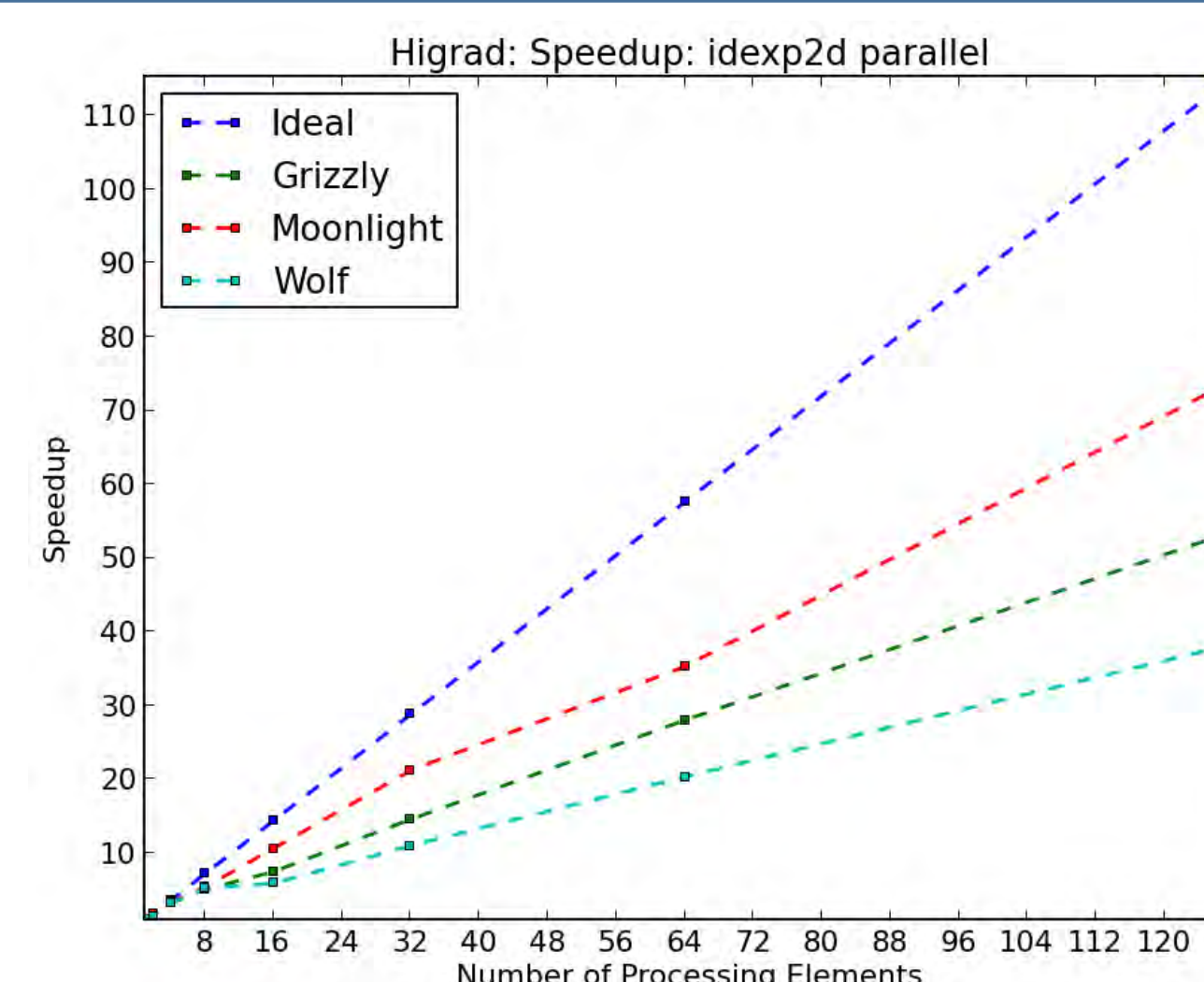


Figure 3: Strong scaling 2D ideal explosion test on different LANL machines

Parallel speedup vs number of MPI ranks for a test problem featuring an expanding semicircular pocket of hot gas. Simulation dimensions are 700x1000. The model is run for 200 time-steps. Shown are curves for three LANL HPC machines:

Grizzly: Intel Xeon Broadwell (2.1GHz, 18 cores)
Moonlight: Intel Xeon E5-2670 (2.6GHz, 8 cores)
Wolf: Intel Xeon E5-2670 (2.6GHz, 8 cores)

This is a surprising result given that one would expect Grizzly to produce the best scaling

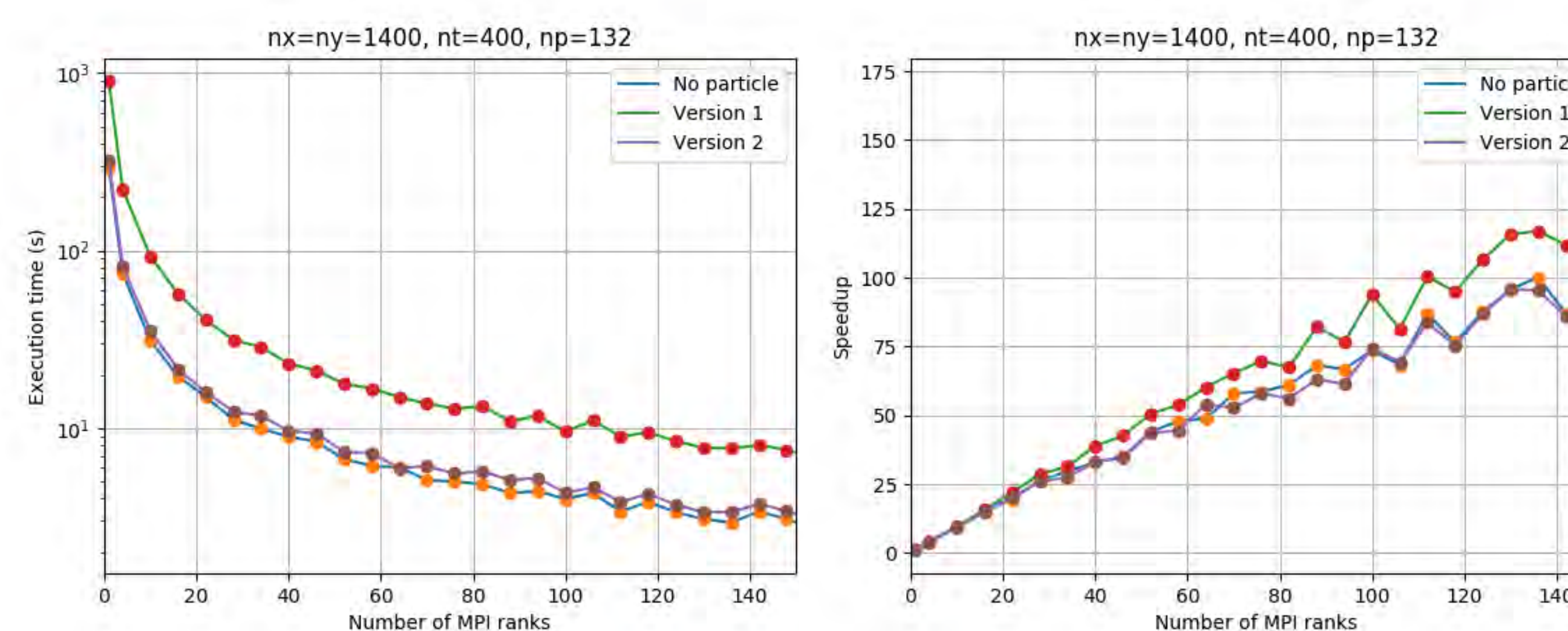


Figure 4: MPI rank scaling test with particles

Graphs of execution time and speedup vs. MPI rank for a test problem featuring 132 particles arranged in a semicircle around an initial pocket of hot gas. Simulation dimensions are 1400x1400, and the model is run for 400 timesteps. Particle version 2 is clearly the most efficient. Each point represents the mean of three runs.

The following `run` command was used to execute these simulations, and was found to produce the best results on Intel Xeon Broadwell nodes

`run --cpu_bind=core --contiguous -n [nranks] [executable] [inputfile]`

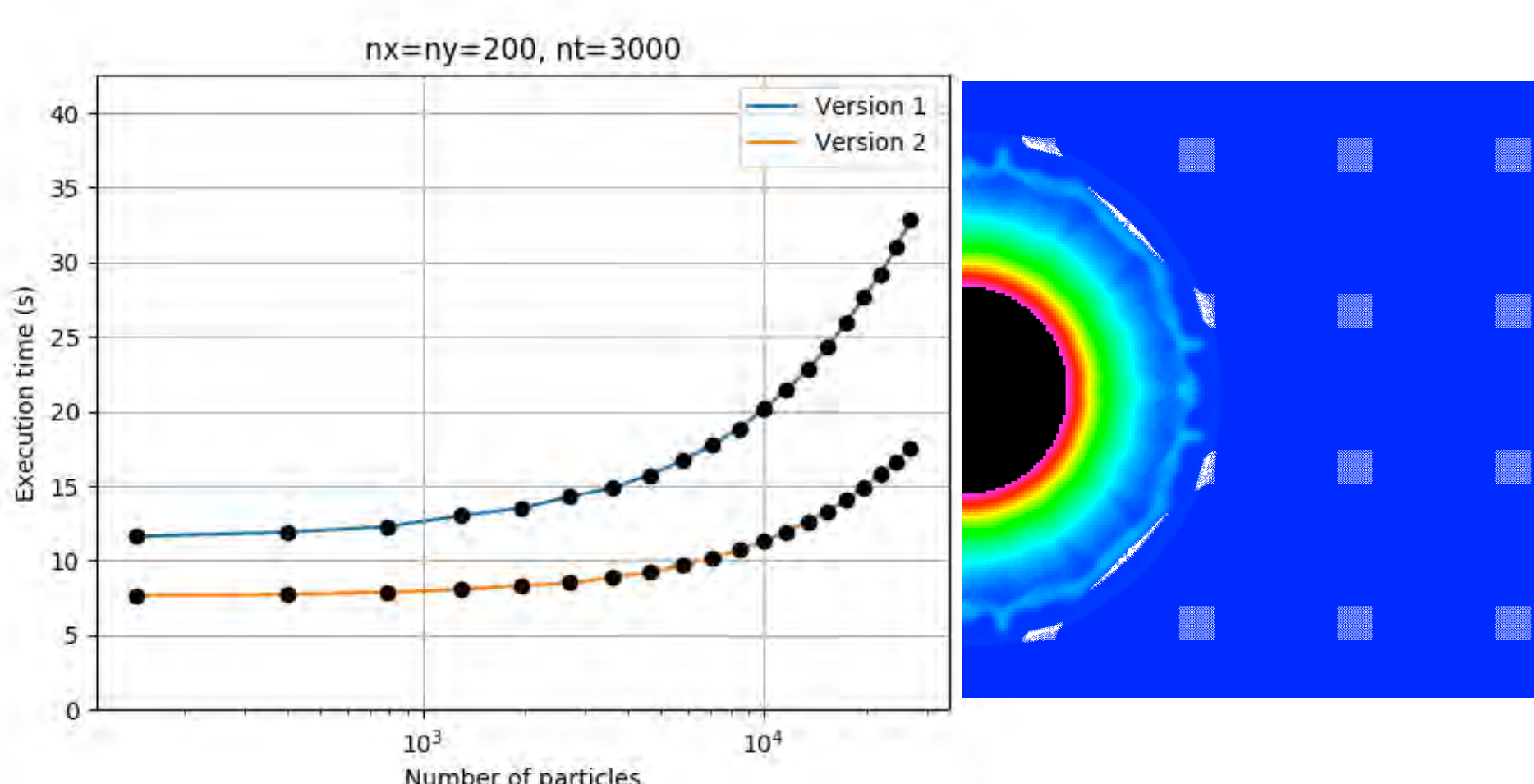


Figure 5: Particle number scaling test Graphs of execution time vs. particle number for the test problem using 16 MPI ranks. Particles are placed in boxes at the center of each rank's domain. The simulation dimensions are 200x200, and the model is run for 3000 timesteps, until the shock front reaches the right wall. Again, particle version 2 is clearly the more efficient.

Real-time visualization

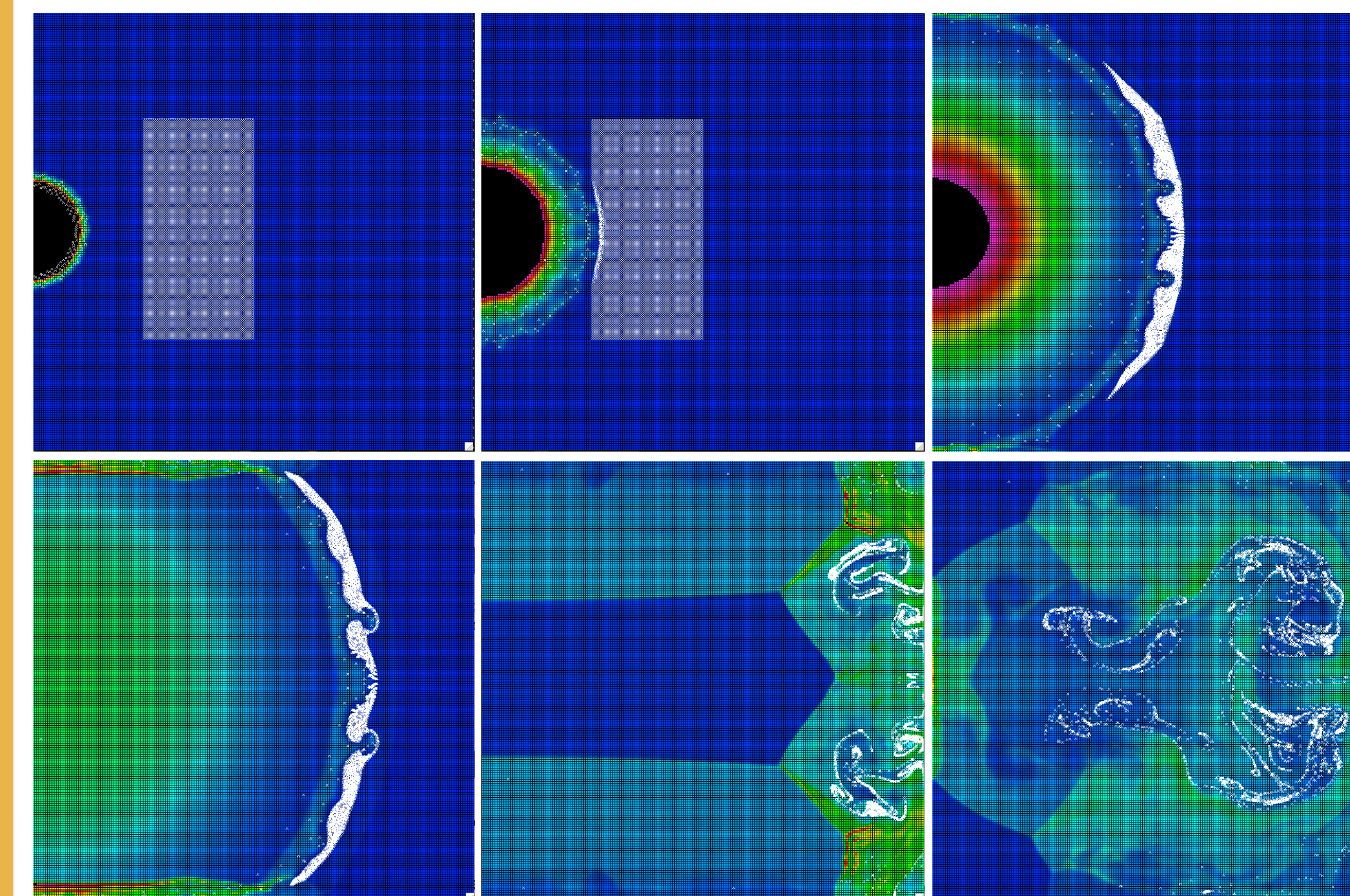


Figure 4: An 'Ideal Explosion' simulation with 2500 particles

A real-time OpenGL graphics package is used to display the simulation results. Here, a semicircular packet of hot, dense gas surrounded by halo of particles three cells wide expands and produces a shock front that interacts with a rectangular block of particles. The domain size is 200x200, timestep 1e-7 s, with no-normal flow boundary conditions. This was run on 16 MPI ranks on Intel Xeon Broadwell processors. Total simulation time of 20,000 timesteps takes about 7 minutes. The particles experience drag forces, which can be customized by the user to reflect differing particles and fluid properties. This real-time OpenGL package provides a vital tool for quick visualization and debugging.

Output is colored by density, while particle positions are marked by white crosses.

Conclusion & Future directions

We have developed two working particle representations and tested them extensively with MPI. The addition of a real-time OpenGL graphics package has provided a useful tool for debugging. Representation of the particles in a 1D array structure appears to be the most efficient option and should be adopted as the main avenue for future development. Possible future directions are as follows:

- High level OpenMP to increase parallelism in particle routine (already started)
- Building a collision model – this will require keeping track of the cell IDs of the particles
- Building an ability to do three dimensional (3D) particle simulations
- Building an easy way for the user to specify particle properties such as size and type. Eventually we would strive towards a system that allows us to simulate a range of particle types simultaneously.

Acknowledgements

This work was supported by the Los Alamos National Laboratory under contract DE-AC52-06NA25396. We benefited from helpful discussions with Jon Reisner, Jesse Canfield and Brian Kaiser. Thanks also to Kris Garrett, Joe Schoonover, Bob Robey and Hai Ah Nam for lectures as part of the 2017 Los Alamos Parallel Computing Internship.

This project was made possible by the computational resources of LANL HPC systems – especially the Grizzly cluster – and the staff members who maintain them.

References

- Cheng, J. & Plassmann, P., 2001, March. The Accuracy and Performance of Parallel In-Element Particle Tracking Methods. In *PPSC*.
- Koo, E., et al., 2012. Modelling firebrand transport in wildfires using HIGRAD/FIRETEC. *International journal of wildland fire*, 21(4), pp.396-417.
- Linn, R.R. E. Koo, J. Canfield, J. Sauer, and J. Winterkamp, "High-resolution atmospheric modeling" prepared for Computational Fluid Dynamics Review, LA-UR-13-23143, Los Alamos National Laboratory, 2013.
- Yeung, P. & Pope, S., 1988. An algorithm for tracking fluid particles in numerical simulations of homogeneous turbulence. *Journal of computational physics*, 79(2), pp.373-416.